VME Readout at and Below the Conversion Time Limit

Citation for the original published paper (version of record):
Munch, M., Jensen, J., Loeher, B. et al (2019). VME Readout at and Below the Conversion Time Limit. IEEE Transactions on Nuclear Science, 66(2): 575-584.
http://dx.doi.org/10.1109/TNS.2018.2884979

VME Readout at and Below the Conversion Time Limit

M. Munch, J. H. Jensen, B. Löher, H. Törnqvist, and H. T. Johansson

***Abstract*— The achievable acquisition rates of modern triggered nuclear physics experiments are heavily dependent on the readout software, in addition to the limits given by the utilized hardware. This paper presents an asynchronous readout scheme that significantly improves the livetime of an otherwise synchronous triggered Versa Module Eurocard Bus-based data acquisition system. A detailed performance analysis of this and other readout schemes, in terms of the basic data transfer operations, is described. The performance of the newly developed scheme as well as synchronous schemes on two systems has been measured. The measurements show excellent agreement with the detailed description. For the second system, which previously used a synchronous readout, the deadtime ratio is at a 20-kHz trigger request frequency reduced by 30% compared to the nearest contender, allowing 10% more events to be recorded in the same time. The interaction between the network and readout tasks for single-core processors is also investigated. A livetime ratio loss of a few percents can be observed, depending on the size of the data chunks given to the operating system kernel for network transfer. With appropriately chosen chunk size, the effect can be mitigated.**

***Index Terms*— Asynchronous, buffering, data acquisition, deadtime, livetime, nuclear physics, performance evaluation, readout, triggers, Versa Module Eurocard Bus (VME).**

I. Introduction

MOST modern nuclear physics experiments have detectors, front-end electronics, computer control, and network. The role of the front-end electronics is to digitize the detector signals such that they can be analyzed. Modular front-end electronics are typically housed in crates such as Nuclear Instrumentation Module, FASTBUS, Computer Automated Measurement and Control (CAMAC), or Versa Module Eurocard Bus (VME), where the latter three also contain a bus on the crate backplane. A typical crate configuration consists of a group of front-end modules together with a controlling single-board computer (SBC). The task of the SBC is to acquire data from the front-end modules and transfer it over the network to permanent storage and online analysis. The speed, overhead, and serialization effects of this transfer naturally limit the maximum achievable acquisition rate, characterized by the rate of accepted triggers.

This paper will focus on VME-based [1] readout systems. The purpose is not to introduce new electronics. Instead, we aim at better utilizing the commercially available modules by generally introducing an extreme asynchronous multievent readout scheme called shadow readout. The main speedup is achieved by almost completely decoupling the readout from the conversion sequence, thereby significantly lowering the readout overhead associated with each accepted trigger. In addition, the coincidence information leading to each trigger is recorded, thus not sacrificing event selection flexibility for speed. It is also shown that the remaining system deadtime can be accurately described based on the timing of basic data transfer operations.

This paper is structured in the following way: first, the existing solutions are reviewed, and the necessary concepts in order to model the deadtime and efficiency of different readout modes are introduced. This is followed by a brief discussion of various modes of VME access. We then describe our implementation and the caveats that arise from having a single-core SBC. Finally, we benchmark the improved readout scheme versus the other strategies on two different systems.

II. Existing Solutions

With CAMAC systems, LeCroy introduced fast readout using the FERA bus in the early 80s, which transfers data at 10 Mword/s (16-bit) [2]. With conversion times, at that time, usually around 3-μs per channel or 25 μs for an eight-channel module, a crate with 50 words of data per event (after zero-suppression) would be read out in 5 μs. Thus, the readout overhead was much smaller than the conversion time.

FASTBUS modules [3], even though known for their high channel density and thus long total module conversion times, usually had multievent buffers, allowing readout of the previous events while converting new ones, completely hiding the readout.

The current state of affairs for VME-based readout systems is that nearly all front-end modules are capable of storing multiple converted events in an internal memory buffer. The Daresbury MIDAS [4] and BARC-TIFR LAMPS [5] utilize these buffers fully to decouple readout from conversion. In this

Manuscript received October 7, 2018; revised November 22, 2018; accepted November 30, 2018. Date of publication January 11, 2019; date of current version February 13, 2019. The work of M. Munch was supported by the European Research Council through ERC Starting Grant LOBENA under Grant 307447. The work of J. H. Jensen was supported by the Danish Council for Independent Research-Natural Sciences under Grant DFF 4181-00218. The work of B. Löher and H. Törnqvist was supported by the GSI-TU Darmstadt Cooperation Agreement. The work of H. T. Johansson was supported in part by the Swedish Research Council under Grant 822-2014-6644 and in part by the Lars Hierta Memorial Foundation.

M. Munch and J. H. Jensen are with the Department of Physics and Astronomy, Aarhus University, 8000 Aarhus, Denmark (e-mail: munch@phys.au.dk).

B. Löher and H. Törnqvist are with the Institut für Kernphysik, Technische Universität Darmstadt, 64289 Darmstadt, Germany, and also with the GSI Helmholtzzentrum für Schwerionenforschung GmbH, 64291 Darmstadt, Germany.

H. T. Johansson is with the Department of Physics, Chalmers University of Technology, SE-412 96 Göteborg, Sweden.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2018.2884979

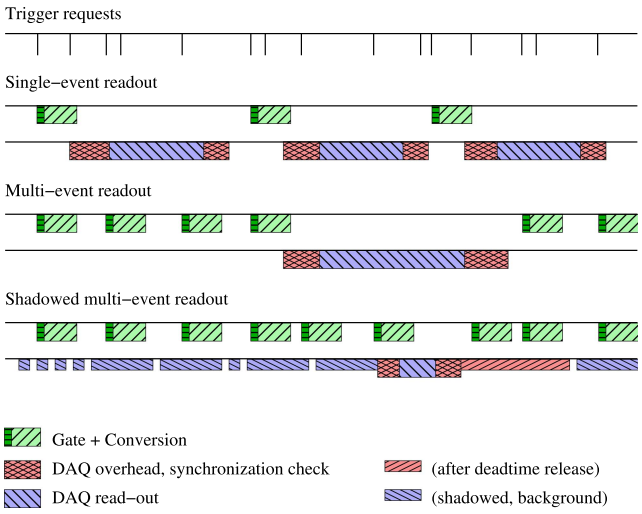

Fig. 1. Single, multievent, and shadow multievent readout. Multievent readout allows event data to accumulate in the module buffers before readout, amortizing the overhead over several events. Shadow readout continuously empties the modules during conversions, thus reducing the work during global deadtime.

scheme, the module buffers are continuously emptied by the crate controller. Only if a module becomes full, will it assert a long busy, halting the acquisition until the SBC empties a part of the module buffer. This mode, however, is only available for the Silena S9418 front-end modules in MIDAS [6], and the LAMPS is hampered by large overhead times in the associated CAEN VME controller [7].

However, the above systems are exceptions, and the multi-event buffers are often not utilized to their full extent, or not at all. Fig. 1 illustrates the three principal modes in which a readout system can be operated when front-end modules have buffers. In the simplest case, the front-end acquires an event and waits for the readout to transfer it. This is called single-event mode. Typically, the data transfer will take significantly longer than the conversion time due to various overheads. These overheads can be partially amortized by filling the front-end buffers before performing a readout. This is called multievent mode. However, the number of events that can be acquired in one go is limited to the shallowest buffer. To the best of our knowledge, common general-purpose nuclear physics data acquisition (DAQ) systems, such as MIDAS (PSI/TRIUMF) [8], MBS (GSI) [9], and RIBFDAQ (RIKEN) [10], use versions of these schemes, where the SBC must interact with the trigger logic for every readout round.

In order to minimize the transfer times, when running in single-event mode, the mountable controller (MOCO) [11] was recently developed at RIKEN. This innovative design is an adapter board with an field-programmable gate array (FPGA) and a USB interface which is installed between a front-end module and the VME crate backplane. With MOCO installed, the data lines of the front-end module are not connected to the backplane. Instead, the FPGA communicates directly with the front-end module. The FPGA can then transfer the data to the controlling computer via USB. The two main benefits of this solution are cost and parallelization of the readout of multiple modules.

In this paper, we will show that a continuous readout mode, operating beyond the depth of the multievent buffers, can be applied generally, without introducing new electronics.

III. Deadtime Modeling

In this section, we develop a framework to describe the deadtime of a system. This model will be used to analyze the systems in Section VII.

Assume that the front-end electronics and readout system is provided with a stream of Poisson-distributed trigger requests, with an average frequency f_r, of which it can accept some frequency f_a. The livetime ratio is the fraction of accepted trigger requests

$$\mathcal{L} = \frac{f_\mathrm{a}}{f_\mathrm{r}} = \frac{1}{1 + f_\mathrm{r}\,\Delta t}. \tag{1}$$

Δt is the amount of time per event that the system is blocked. The last equality assumes the deadtime to be nonextending, and the formula is derived in [12]. The livetime ratio is related to the deadtime ratio via $\mathcal{D} = 1 - \mathcal{L} = f_\mathrm{a}\,\Delta t$.

We carefully differentiate between a system in deadtime and the deadtime ratio $\mathcal{D}$. A system in deadtime rejects events while the deadtime ratio is the fraction of rejected events. The same differentiation is made between the livetime and the livetime ratio, $\mathcal{L}$.

It is often necessary to consider a distinct contribution to the total system deadtime: busy. Busy signals come from front-end modules, and is also autonomously released by those. It is generally asserted during gate and conversion, and when the data buffer of a module is full. The more regular deadtime is initiated by the trigger logic and removed by the SBC readout software. As far as trigger acceptance is concerned, both have the same effect: inhibit triggers.

While running, the system must perform the following tasks (with associated processing times):

1) apply the gate of each accepted event, t_g;
2) convert the event, t_c;
3) poll for data, t_p (per poll);
4) readout overhead (e.g., determine data amount), t_o;
5) read data, t_d;
6) check module synchronicity, t_s;
7) transfer data over the network, t_n.

The two first are handled by the front-end electronics, the other by the SBC. Some of the tasks happen during either live or deadtime, depending on the design of the system. Thus the minimal deadtime ratio is given by the fraction of time modules are busy converting data

$$\mathcal{B} = f_\mathrm{a}(t_\mathrm{g} + t_\mathrm{c}). \tag{2}$$

If the system is designed such that it reads one single event at a time and will not accept new events while reading data, then the readout time is $(t_\mathrm{p} + t_\mathrm{o} + t_\mathrm{d} + t_\mathrm{s})$, since the system must poll for data, determine the amount of data, read the data and check for synchronicity. Since the likelihood that a random trigger request occurs during deadtime is simply the total fraction of time the system will not accept triggers, the deadtime ratio is

$$\mathcal{D} = \mathcal{B} + f_\mathrm{a}(t_\mathrm{p} + t_\mathrm{o} + t_\mathrm{d} + t_\mathrm{s}). \tag{3}$$

However, this scheme suffers the overhead of $t_p + t_o + t_s$ for every event. If the front-end electronic modules have buffers of a certain depth n_b, then the cost of polling, general readout overhead, and synchronization can be amortized

$$\mathcal{D} = \mathcal{B} + f_a \left(t_d + \frac{t_o + t_p + t_s}{n_b} \right). \tag{4}$$

This mode of operation is commonly referred to as multievent mode with the former called single-event mode. In this case, t_d can often also be reduced as the larger amounts of data per transfer may profit from faster transfer modes.

If, however, the system can simultaneously accept new events and transfer data, then $\mathcal{D}$ may be further reduced

$$\mathcal{D} = \mathcal{B} + f_a \frac{\alpha t_d + t_o + t_p + t_s}{n_s}. \tag{5}$$

n_s is the number of events accepted before synchronicity is checked and α is the average number of events remaining to read out during deadtime. The fraction of events read during deadtime is thus α/n_s. $\alpha \geq 0$ but will typically only be a few events. It should be noted that n_b for many older modules is typically 32 or less, while newer modules might store a few hundred events. However, n_s can in principle be made arbitrarily large, and thus the deadtime can be reduced to the busy time, $\mathcal{D} \approx \mathcal{B}$.

The above has not considered the available data transfer bandwidth, which may also limit the maximum event rate that can be accepted

$$f_{a,\max} = \frac{1}{(t_d + \frac{t_o + t_p}{n_b} + \frac{t_s}{n_s})}. \tag{6}$$

In practice, the achievable frequency will be slightly lower due to various other overheads such as network transfer. Above this request frequency, the livetime ratio just deteriorates while no more events are accepted and is simply described

$$\mathcal{L} = \frac{f_{a,\max}}{f_r}. \tag{7}$$

The livetime of the system is thus given by whichever of (1) or (7) is smaller

$$\mathcal{L} = \min \left(\frac{1}{1 + f_r \Delta t}, \frac{f_{a,\max}}{f_r} \right). \tag{8}$$

For shadow readout, Δt is expected to be approximately equal to the gate and conversion time $t_g + t_c$. The livetime characteristics of different readout schemes are illustrated in Fig. 2, where also the hard limit nature of $f_{a,\max}$ is clearly seen.

In practice, the transition between the two domains is not sharp, due to interplay among different saturating mechanisms, which is not investigated further. However, a smoothed minimum function is needed when fitting measured behavior in Section VII. As the exact nature of the smoothing is not considered important, different functions could be used. In this paper, the following function has been used, since it only depends on a single parameter k

$$\min{}_k(x, y) = -\frac{1}{k} \ln \left(e^{-kx} + e^{-ky} \right). \tag{9}$$

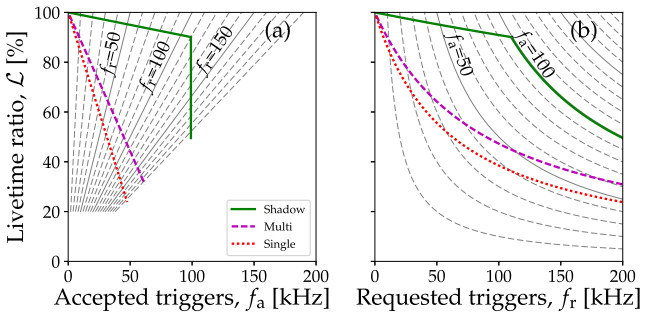

Fig. 2. Schematic performance of shadow, multievent, and single-event modes, with $t_g + t_c = 1\ \mu s$, $t_p + t_o = 3\ \mu s$, $t_d = 10\ \mu s$, $t_s = 2\ \mu s$, $n_b = 32$, and $n_s = 1000$. Both figures show the same systems (a) as a function of f_a and (b) as a function of f_r. The background lines show constant f_r and f_a, respectively.

IV. VME Transfer Time

The time it takes for the basic operation of transferring a 32-bit data-word essentially determines t_p, t_o, t_d, and t_s. Generally, VME access can be divided into two categories, namely, single cycle (SiCy) and block transfer, where the latter can transfer either 32 (BLT) or 64 (MBLT) bits at a time [1]. There is also 2eVME and 2eSST [19] (both are block transfer modes), but these are not widely supported in front-end modules.

SiCy transfers single data words with minimal overhead. Thus with module i producing d_i words, the data transfer time with SiCy reads is

$$t_d = \sum_i d_i t_{i,\mathrm{SiCy}}. \tag{10}$$

Since the designs of the individual modules also affect the word transfer times, t_{SiCy} has a module dependence, $t_{i,\mathrm{SiCy}}$. While block transfers are asymptotically faster per word ($t_{i,\mathrm{block}} < t_{i,\mathrm{SiCy}}$), they may require significant setup times in the SBC

$$t_d = \sum_i \left(t_{\mathrm{setup}} + d_i t_{i,\mathrm{block}} \right). \tag{11}$$

It is, however, possible to amortize this setup time by performing a so-called chained block transfer, whereby data is read from multiple modules during the same transfer[1]

$$t_d = t_{\mathrm{setup}} + \sum_i d_i t_{i,\mathrm{block}}. \tag{12}$$

With c_i being the number of reads required to determine the amount of data and ensure synchronicity for each module, $t_o + t_s$ can be expressed analogous to (10)

$$t_o + t_s = \sum_i c_i t_{i,\mathrm{SiCy}}. \tag{13}$$

So far the discussion has been general. However, to evaluate actual systems, it is necessary to measure the transfer times.

[1]Note that chained block transfers are not an additional VME transfer mode, but something some modules can be configured to arrange together, e.g., by using the IACK backplane chain for communicating a token between the modules.

TABLE I
TRANSFER TIME OF TWO DIFFERENT SBCS WITH VARIOUS MODULES

SBC	Module	SiCy	SiCy DMA	BLT	MBLT
MVME	DMA setup	-	16.7 ($n = 1$)		
	MTDC-32	1.21			
	MADC-32	1.26	0.49	0.44	0.33
	VULOM4b (TRLO II)	1.37	0.65		
RIO4	(M)BLT setup	-	-	$6.5 + n \cdot 4.3$	
	MTDC-32	0.40		0.17	0.09
	MADC-32	0.45		0.22	0.12
	VULOM4b (TRLO II)	0.60			
	V785	0.50		0.19	0.15
	V1190	0.45		0.18	0.10

Table I shows the measured transfer times in µs per 32-bit word for two different SBCs and various modules. The additional SBC setup overhead of starting DMA (direct memory access) or block transfers depend on how many, n, are scheduled at the same time. DMA allows multiple adjacent SiCy requests to be scheduled together on the MVME. The MTDC-32 [13] and MADC-32 [14] are from Mesytec, the V785 [15] and V1190 [16] from CAEN, and the VULOM4b [17] from GSI and is running the TRLO II firmware [18].

Table I shows the measured read transfer time for a selection of commercial modules when read using either a Motorola[2] MVME5500 [20] (MVME) or a CES[3] RIO4-8072 [21] (RIO4) SBC. Both are operated with a Linux kernel [22], [23], versions 2.4.21 and 2.6.33, respectively. These transfer times will be used in Section VII when modeling readout systems with these components.

Comparing the MVME and RIO4 SiCy transfer times in Table I, the RIO4 is generally three times faster per word. Note that the timing differences between modules for the same SBC are roughly equal for the two SBCs. This reflects the fact that it is the module VME access handling that determines these differences.

The measurements also clearly show that (M)BLT is significantly faster per word than SiCy, but has a considerable setup overhead.

V. IMPLEMENTATION

In Fig. 3, the work performed by the readout loop is sketched as a flow diagram. Generally, the SBC will either poll a register or be notified via an interrupt that a readout should be performed. The trigger logic has then asserted deadtime. When the readout request is found, the SBC will check all module event counters and move the remaining data from all front-end modules to a CPU buffer. Then, it can release the deadtime, allowing the front-end modules to acquire more data while the SBC performs any further consistency checks of the acquired data. Afterward, it will resume waiting for the next readout request.

The difference between multievent and shadow readout is the activities taking place while waiting for a readout request.

[2]Motorola's Embedded Communications Computing business was acquired by Emerson in 2007.

[3]CES was acquired by Mercury Systems, Inc. in 2016.

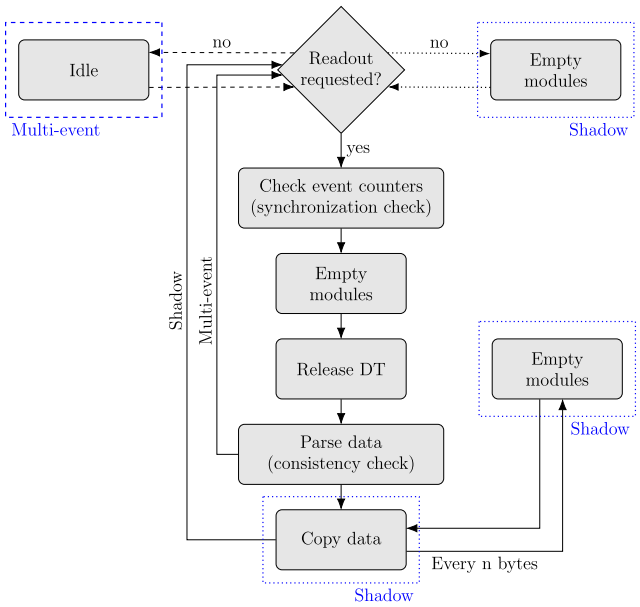

Fig. 3. Flow diagram for multievent and shadow readout. In multievent mode, idle time is spent doing nothing or work for other processes. In shadow mode, partial module readout is continuously performed. In both cases, the main readout will look for a readout request, which has asserted deadtime and will then transfer the (remaining) data and ensure that all modules are in sync. Then, deadtime is released. Since data copy in shadow mode may be a lengthy process, the modules are regularly emptied during copying.

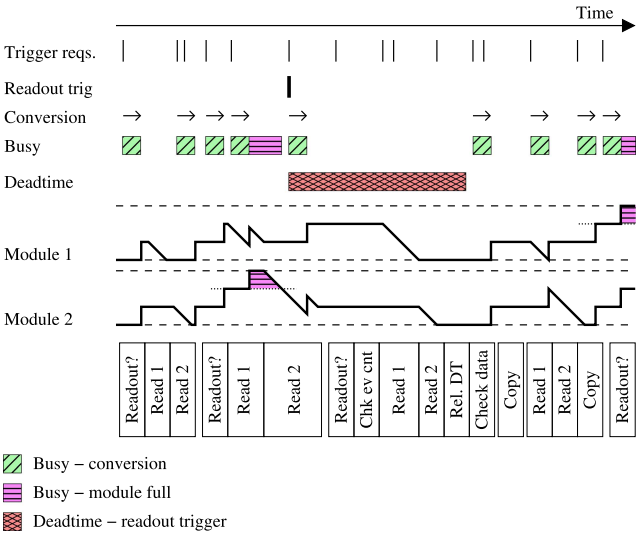

Fig. 4. Example of SBC activity during shadow readout. The SBC repeatedly empties module buffers. Thus, when entering deadtime, the modules will be almost empty. While deadtime is asserted, event counters will be checked and the little remaining data transferred. After deadtime has been released, data will be consistency-checked and copied to the actual output buffer. The module graphs show their buffer fill levels. Busy is asserted both due to gate+conversion, and when any module buffer is full (above dotted lines). In this example, the module buffers can hold three events.

In the multievent case, the SBC is mostly idle, except for network transfer tasks.

The shadow readout on the other hand tries to continually transfer data from the front-end modules. This is illustrated for a two-module system in Fig. 4, which also shows the

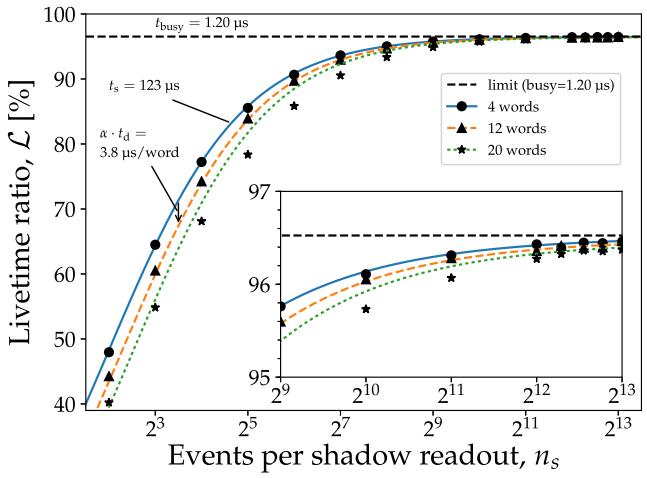

Fig. 5. $\mathcal{L}$ dependence on the number of events per shadow readout for an MVME and MADC-32 system with $t_g + t_c = 1.2\,\mu s$ and $f_r = 30\,\text{kHz}$. $\mathcal{L}$ converges to the conversion limit (dashed horizontal line) when n_s is increased. With lower n_s, the overhead and synchronization time is amortized over fewer events. With larger event sizes (varying d_i), the amount of remaining data to read during deadtime after each n_s block of events increases. The origin of the discrepancies for $d_i = 20$ words and $n_s \geq 2^4$ is unknown.

TABLE II
CPU TIME OF NETWORK CALLS

SBC	Buffer	Network CPU time
MVME5500	32Mi	$1.4\,\mu s + w \cdot 8.8\,ns$
	64ki	$1.1\,\mu s + w \cdot 3.3\,ns$
RIO4	32Mi	$4.5\,\mu s + w \cdot 9.5\,ns$
	64ki	$2.3\,\mu s + w \cdot 3.8\,ns$
E3-1286v6 (Intel x86, 4.5 GHz)	32Mi	$0.41\,\mu s + w \cdot 0.091\,ns$
	64ki	$0.37\,\mu s + w \cdot 0.065\,ns$

Table II shows the CPU time per network `write` call of w bytes for two different SBCs, and a modern server CPU for reference. The values are averages for either a large or a small buffer, the latter typically fitting in low-level CPU cache. The former are representative for DAQ network transfers from large event accumulation buffers.

pausing when a module buffer becomes full and the deadtime management.

In the simplest case, the data from each module are transferred to a separate buffer. This avoids potentially fragmenting events, and the collected data can easily be consistency checked. In this fashion, it is no longer needed to limit n_s to the module buffer depth n_b. Instead, the limitation for n_s is the available RAM and the desire to perform regular synchronicity checks. The dependence on n_s is illustrated in Fig. 5, where measured data are shown with point-like markers, and general trends of models are shown as curves. This scheme is used throughout this paper. Fig. 5 shows $\mathcal{L}$ for a system which has a conversion time $t_c = 1.2\,\mu s$ (dashed horizontal line) and produced $d_i = 4$, 12, or 20 words per event with $f_r = 30\,\text{kHz}$. Fig. 5 shows two essential features of the shadow readout: $\mathcal{L}$ converges toward the limit set by conversion times when n_s is increased. In addition, $\mathcal{L}$ only shows a weak dependence on event size with less than 0.5% variation in $\mathcal{L}$ when $n_s = 8\,\text{k}$.

The virtual module buffers must eventually be copied to an output buffer. This is done after the consistency check as shown in the flow diagram. However, since a substantial amount of data may have been collected, this can easily take several milliseconds, which may be longer than module buffers can continue to store new events, causing them to assert busy signals. In order to avoid this, each module is assigned two memory buffers operated in a "ping-pong" fashion such that new data is transferred into the currently active buffer. When the modules have been emptied during deadtime, the other buffer becomes active and will be filled with new data. The data from the inactive buffer is copied in chunks interleaved with frequent calls to the shadow readout routine. This is illustrated with the read of the modules in-between the copy blocks in Fig. 4.

We have implemented this readout scheme with a modified version of the readout library nurdlib (formerly known as vmelib) [24]. When the readout polling is performed by the surrounding DAQ framework code, it is necessary to also modify that. The change essentially amounts to one single line of code—a callback routine to allow data transfers in the background while no readout request is detected.

Hardware requirements are the same for multievent and shadow readout, i.e., there must be an acquisition control module that rejects events whenever deadtime or busy is asserted. This module, which issues the readout requests, must be able to keep track of how many events have been acquired and only request a deadtime-asserting readout after a sufficient number (n_s). In our setup, all these tasks are performed by the TRLO II firmware [18] running on the GSI VULOM4b module [17].

To allow the most flexible use of the data from a multievent readout system, it is necessary also to be able to record for each trigger the detector coincidences that caused the trigger to be selected. This trigger coincidence pattern is recorded for each event by the TRLO II firmware. In order to trust the system, it is also necessary to verify that each trigger has been seen by each module once (none lost, none spurious). This is done by comparing the event counters in the modules regularly, i.e., during each deadtime period. This is enough since a correctly working system would never have mismatches, making any deviation significant. Nurdlib already performs these checks strictly.

Note that shadow readout causes almost continuous activity on the VME backplane during analog conversion. Within our tests, it was no problem, provided that the electronic modules and preamplifiers etc. were properly grounded. It is suggested to compare the noise levels with and without shadow readout, e.g., using deadtime-asserting multievent readout for the latter.

VI. SINGLE-CORE CAVEAT

Both the MVME and RIO4 have single-core processors, which poses some challenges related to the kernel scheduler of the operating system. The main issue is the conflict between readout and network transfer. The SBC must transfer the data to either an event builder or nonvolatile storage, meaning time not spent emptying modules (see Table II). It will, therefore, lower the maximum rate that can be handled. However, before

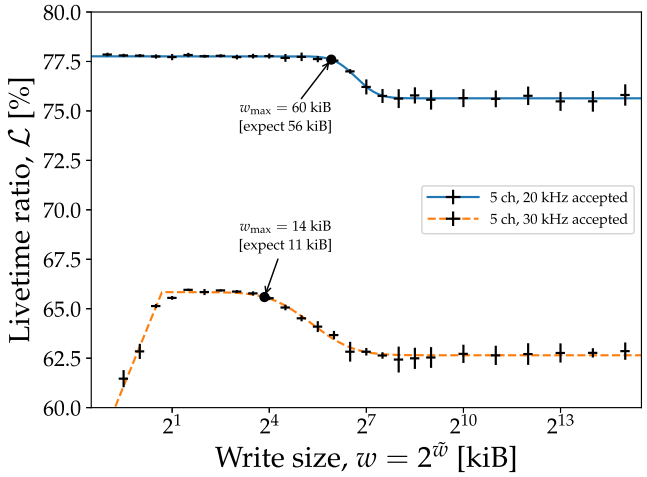

Fig. 6. $\mathcal{L}$ as a function of network preparation chunk size w, for two different average trigger rates. The system consists of a RIO4 with a VULOM4b and six CAEN V785s, delivering 33 words per event (five channels per ADC). When the chunk size exceeds $w_{\max}$, the livetime ratio drops due to module buffers occasionally becoming full while the CPU is occupied with network processing. Also, too small chunk sizes cause losses, due to CPU time lost on performing unnecessarily many calls.

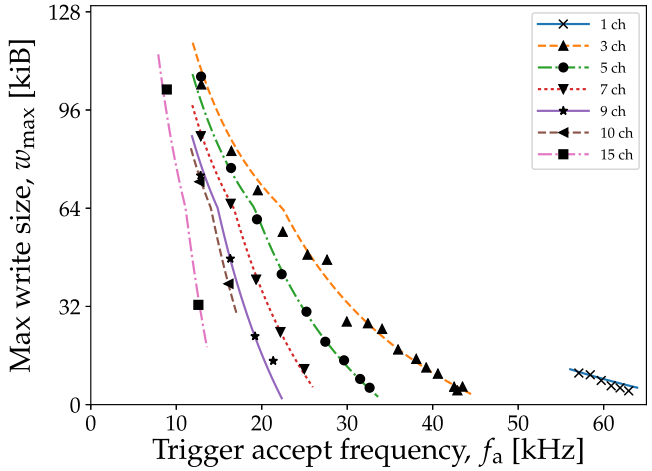

Fig. 7. Maximum chunk write size $w_{\max}$ for the system in Fig. 6, determined for many combinations of event sizes and trigger rates. The event sizes are varied by activating a different number of channels in the ADCs.

that it can also impact the livetime of the system, if the network preparation happens in such large uninterrupted chunks that the module buffers become full. This is shown in Fig. 6, where $\mathcal{L}$ drops by a few percents in the naive case where the network transfer is issued with too large buffers sent to the `write` system call. The kernel takes too long to complete the request, which causes the drop in $\mathcal{L}$. This can be mitigated by breaking the transfer preparation into multiple chunks, each with a call to `write` directly followed by yielding the timeslot of the network thread (by `sched_yield`). This allows the readout thread to cycle through the modules for each `write`. The drop in $\mathcal{L}$ can be described by considering the maximum time each network preparation chunk should take

$$t_{\mathrm{n,max}} = \beta \frac{n_{\mathrm{b}}}{f_{\mathrm{a,max}}} - \left(t_{\mathrm{d}} + \frac{t_{\mathrm{o}}}{n_{\mathrm{b}}} \right). \tag{14}$$

The first term is the average time the module buffer can handle, and the second term the CPU time spent on the readout. The factor β accounts for the fact that a Poisson-distributed trigger will sometimes issue a burst of many triggers in an unusually short time interval than the long-term average, and thus fill the buffer more quickly.

The maximum time can, by using the values of Table II, also be expressed as a maximum chunk size, which can be used to control the network processing

$$w_{\max} = 2^{\tilde{w}_{\max}} = \frac{t_{\mathrm{n,max}}}{t_{\mathrm{write/B}}}. \tag{15}$$

The drop in the livetime ratio as w exceeds $w_{\max}$ (as shown in Fig. 6) can be fitted by

$$\mathcal{L}_1 = \mathcal{L}_{\max} - \mathcal{L}_{\mathrm{drop}} \left(\frac{1}{2} + \frac{1}{2} \mathrm{erf} \frac{\tilde{w} - \tilde{w}_0}{\tilde{w}_\sigma} \right). \tag{16}$$

We use the "error function" (erf) to describe the slight but measurable drop in $\mathcal{L}$ around `write` size $2^{\tilde{w}_0}$.

Using too small chunk sizes, however, will introduce a steep loss, due to the overhead of very many `write` calls. This is seen for the 30-kHz case in Fig. 6.

The above description has been checked for many combinations of trigger frequencies and event sizes, as presented in Fig. 7. For each case, $w_{\max}$ has been determined from fits of (16) as in Fig. 6 with the value $2^{\tilde{w}_{\max}} = 2^{\tilde{w}_0 - \tilde{w}_\sigma}$. The curves thus describe estimates of the maximum `write` size that can be used without affecting performance. At buffer sizes above 64 kiB, the fit assumes that the network overhead is twice as large, matching the measured data. We have, however, not been able to attribute this effect to a specific cause. The best fit value of $\beta = 0.79$ corresponds to bursts happening 10%–1% of times for f_{a} in the range of 10–50 kHz. The percentages were obtained from simulations of Poisson-distributed triggers taking a nonextending 10-μs busy-time into account for stretches of $n_{\mathrm{b}} = 32$ accepted triggers.

Even if the negative impact on the livetime is only a few percents, mitigation is important since the effect is enhanced for multinode systems as the trigger requests which become blocked on each node are independent. This cumulative effect is illustrated in Fig. 10(b).

Provided that any common data bus between the CPU and the network and VME interfaces can handle the simultaneous traffic, this effect could be mitigated with a multicore SBC. One core would be dedicated to the readout, while another would handle noncritical tasks such as network transfers.

VII. System Characterization

In this section, we will benchmark the shadow readout system versus a regular multievent readout system using two different sets of modules. Each set is part of actual experiments. The first system was used for the IS633 experiment at ISOLDE (CERN) [25] and the second one is the current system used at the Aarhus 5-MV accelerator.

In all cases, the trigger requests are either provided by a CAEN DT5800D detector emulator (pulser) [26], which can provide a Poisson distributed trigger sequence with a given

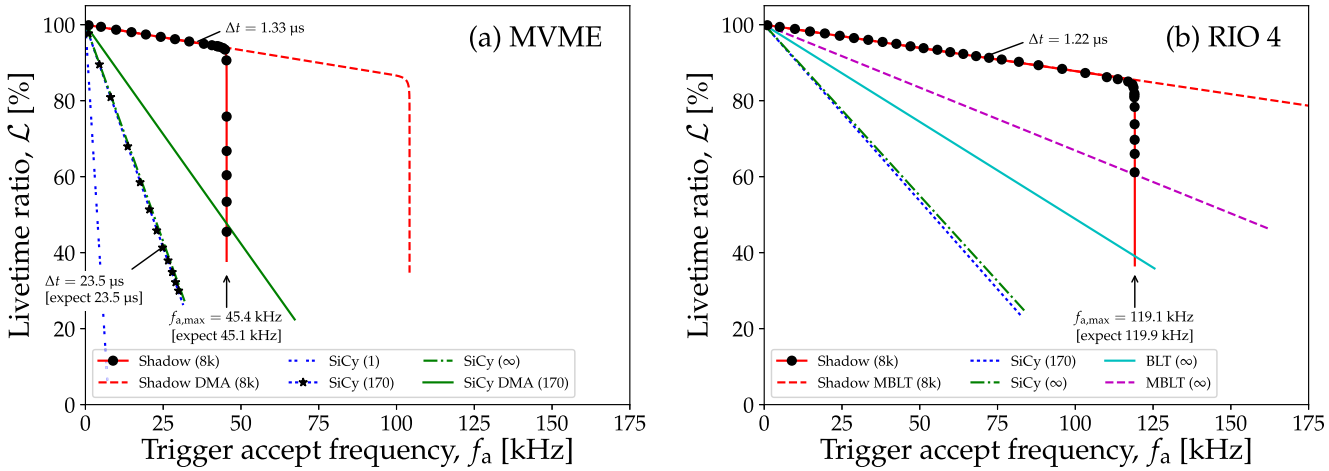

Fig. 8. Livetime ratio $\mathcal{L}$ as a function of accepted trigger frequency for the MVME and RIO4 SBCs reading from the VULOM4b, MADC-32, and MTDC-32 using various readout modes, with in total 17 words per event. The round data points show the measured $\mathcal{L}$ for shadow readout, while the stars are for multievent mode, both with SiCy readout. The numbers in parentheses are the number of events accepted before the SBC readout function is invoked. ∞ corresponds to the limit of zero overhead. Note that the lower performance of the MVME SBC (see Table I), only affects the maximum number of events taken, $f_{\mathrm{a,max}}$, not the per-event deadtime at lower rates.

rate, or by equivalent functionality directly in the trigger logic firmware. The trigger requests are sent to the VULOM4b running the TRLO II firmware, which handles the busy and deadtime logic. It also provides scaler values for the total number of trigger requests and the number of accepted requests.

Throughout this paper, measured data are shown with point-like markers, and general trends of models are shown as curves. For illustration, some models are also evaluated for configurations that have not been tested.

A. IS633 System

This is a simple configuration designed to run with very low deadtime. It consists only of an SBC, VULOM4b, Mesytec MADC-32, and Mesytec MTDC-32. In order to achieve high livetime, the MADC is configured in bank toggle mode [14]. In this mode, the MADC toggles between which of its two analog-to-digital converters (ADCs) that digitize the signals and can thus accept a new event while still processing the previous one. The gate was set to 1 μs, and the module in 4 k mode which has a conversion time of 1.6 μs. The module will thus only assert busy if three events arrive within a 2.6$-\mu$s window. However, in practice, pileup is best avoided, and thus TRLO II emitted a 1-μs busy after every accepted event.

The modules were configured such that the VULOM4b produced 3 words of data per event, the MADC-32 12, and the MTDC-32 2. The busy was a logical OR between the three modules. The modules could store 170, 481, and 2891 events, respectively. An additional 66 words (mainly scaler values) were produced and read once per readout request.

1) Multievent Mode: Fig. 8 shows the livetime ratio as a function of the trigger request frequency when using either the MVME or RIO4. For the MVME, $\mathcal{L}$ has been measured for multievent mode using SiCy readout and a buffer depth of 170. The curve through the data points is (1) with $\Delta t = 23.3\,\mu s$. This corresponds to an effective $t_{\mathrm{c}} = 1.3\,\mu s$, the expected $t_{\mathrm{d}} = 21.5\,\mu s$, and a combined overhead per event of $0.53\,\mu s$. The green dashed-dotted curve shows the expected behavior with zero overhead (∞ multievents) and it would only provide a slight improvement since the data transfer dominates the deadtime (curve overlaps with SiCy(170) in Fig. 8). The RIO4 system would perform significantly better with a SiCy readout time per event of only 8 μs. Fig. 8(b) also shows the curves corresponding to 32- and 64-bit block transfer in the limit of zero overhead per event. Using block transfer, the per-event readout time would be reduced to 4.1 or 2.3 μs, respectively.

2) Shadow Mode: Fig. 8 also shows the measured $\mathcal{L}$ for shadow readout doing SiCy reads with a shadow buffer depth of 8192 events. Up to $f_{\mathrm{a,max}}$, the data points follow (1) with Δt roughly equal to the gate time. According to the models discussed earlier [(6) together with values from Section IV], the maximum accept frequency the MVME can sustain is 46.4 kHz while the RIO can handle 125 kHz. Note that up to the respective $f_{\mathrm{a,max}}$, the MVME and RIO4 deliver essentially the same livetime performance.

If the maximum bandwidth of SiCy transfers becomes the limiting factor, one could combine block transfer with shadow readout. The red dashed curves labeled Shadow DMA or MBLT in Fig. 8 show such configurations. In this case, the limiting factor is the block transfer overhead, with an expected $f_{\mathrm{a,max}} = 388$ kHz for the RIO4 with this particular setup and number of channels.

B. Aarhus System

The other system consists of a RIO4, VULOM4b, and six CAEN V785 ADCs. This system is used at the Aarhus University 5-MV accelerator. To mimic actual production conditions, each ADC module was configured to produce five words per event while the VULOM4b produced 3. In total 33 words were produced per event with an additional 66 words produced once per readout.

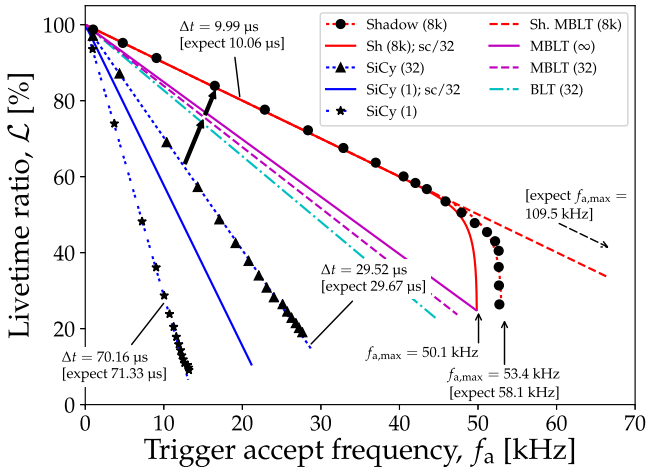

Fig. 9. Livetime ratio $\mathcal{L}$ as a function of accepted trigger frequency for a RIO4 with a VULOM4b and six CAEN V785s for various readout modes. Each event consists of 33 words. The round data points show the measured $\mathcal{L}$ for shadow readout. The numbers in parentheses are the number of events accepted before the SBC readout function is invoked. ∞ corresponds to the limit of zero overhead. sc/32 denotes that scaler data is recorded every 32 events, providing a fair comparison with multievent readout. The values displayed are from fits, while the expected values are predictions based on the values in Table I. The thick arrows indicate the improvement obtained over the previously used multievent mode at working conditions of 20 kHz requested trigger rate. The deadtime ratio is halved and the accepted and thus recorded trigger rate increased by 30%.

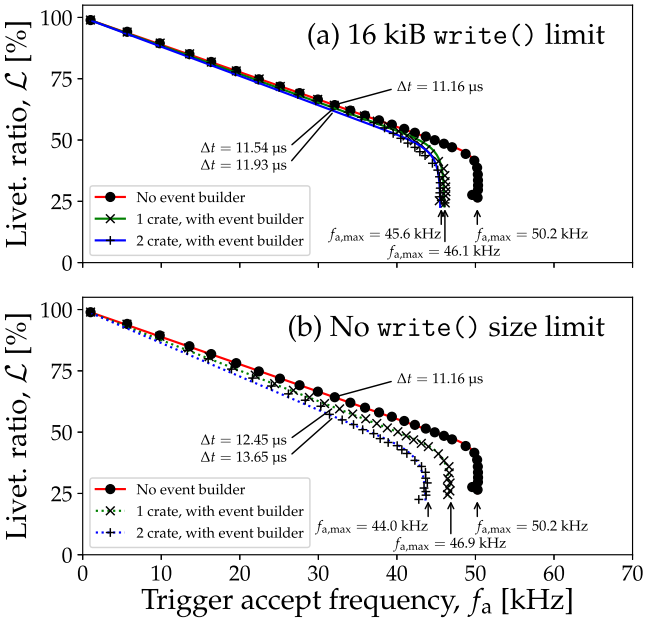

Fig. 10. Livetime ratio $\mathcal{L}$ as a function of accepted trigger frequency for the same system as in Fig. 9, with network transport of data to a common event builder (cross markers), and a multicrate system with two duplicate crates (plus markers). The measurement without network transport (circles) is also shown. (a) Network `write` calls are limited to 16 kiB. Note that before $f_{a,max}$ is reached, $\mathcal{L}$ is in all cases essentially given by $\mathcal{B}$, i.e., it is virtually unaffected by the network transport. (b) Network `write` calls have no limit and thus cause additional deadtime due to module buffers becoming full, while the CPU is busy with network processing. The additional deadtime occurs for independent events, accumulating the effect for multicrate systems.

The CAEN V785 has a 32-entry multievent buffer and a total conversion time of 7.06 μs, which includes settling times etc. In addition, a 3 μs gate is used. Hence, the expected deadtime per trigger is 10.06 μs. It should also be noted that chained block transfer is not advantageous for the V785, as it only can deliver one event per transfer for these modules. Since we have more events than modules, we will instead model the simultaneous scheduling of six block transfers. This has an overhead of 32 μs, i.e., 1 μs per event.

In practice, readout with (M)BLT of the CAEN V785 is troublesome with some modern SBCs, since the data length cannot be queried beforehand and instead is marked with a VME `BERR*` signal. While allowed by the standard, this interacts badly with some SBC block transfer drivers since they do not report the number of actually transferred words, and thus obliterates much of the benefits. In the model, we have assumed that the SBC driver provides the needed information.

Another issue with (M)BLT of the CAEN V785 is that the busy release after conversion is delayed until the ongoing VME transfer has completed [27]. This will inflate the effective conversion time. This effect has also been ignored in the model.

Fig. 9 shows the results obtained for the shadow readout and the measured and estimated behavior of various multievent modes. The SiCy data transfer time per event is 17 μs, while BLT and MBLT take 6.3 and 5 μs, respectively. This includes the block transfer overhead of $\sim$ 15% compared to the transfer time. However, even in the limit of zero overhead, block transfer does not converge to the limit of conversion and gate time. On the other hand, the data obtained show that SiCy shadow readout can maintain the limit and keep up with the data rate until $\sim$ 53 kHz, and to $\sim$ 50 kHz when scaler readout for every 32nd event is included.

C. Network Impact

The above measurements were done without network transport, in order to simplify the system descriptions. The data rates, at a few megabyte per seconds, do not use any significant CPU resources for network processing. This is shown in Fig. 10(a) for a single-crate system with and without network transport, where the network overhead due to sending 6.3 MB/s in 16 kiB chunks should use 6.1% CPU time according to Table II. The lowering of $f_{a,max}$ from 50.2 to 46.1 kHz corresponds to 8.2%.

D. Multicrate System

The shadow readout mode can also be applied to multicrate systems. The Aarhus system described above (RIO4, VULOM4b, and six CAEN ADCs) was duplicated in a second crate, and is operated together with the first in a master-slave configuration. The data from both systems are sent to an event builder PC for merging. The master start signal is used to generate gates for the modules in both crates for each trigger. The master crate generates the readout triggers, and by means of a TRIVA [28] mimic connection [18] they are distributed to be handled simultaneously by the slave system. The readout deadtime is the logical OR combination of the deadtime in the

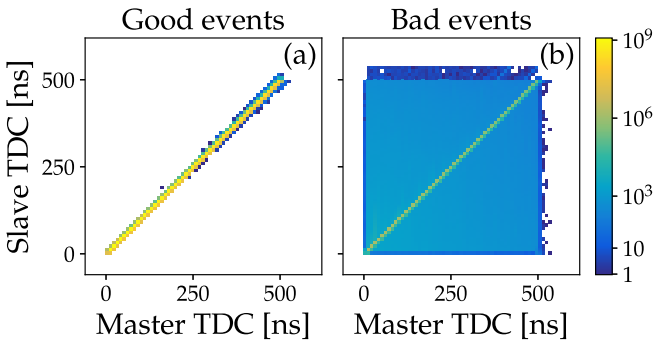

Fig. 11. Shown is the simultaneous measurement of $3.1{\times}10^{10}$ events with the same signal in both the master and slave system. Here, the slave system has in addition to the correct common triggers, shown in (a), also intentionally received a low rate of spurious, i.e., wrong, master starts. This gives rise to $4.2{\times}10^{7}$ diagnosed bad events, collected in (b). The clean diagonal line in (a) demonstrates the ability to detect and discard spurious triggers in a multicrate system. Potentially bad events are, within each shadow readout block of events, detected by event count mismatches between the data from the systems.

two systems. The busy signals are handled similarly, i.e., busy reported by any module in the total system inhibits further triggers.

Fig. 10(a) shows that the trigger and shadow readout operation is virtually unaffected by the second crate provided that the execution time of each network `write` call is limited. Fig. 10(b) shows how stalls due to too long network processing that happen at independent events cause losses that scale with the number of involved readout nodes.

E. Multicrate Event Correlations

In any multievent readout scheme, event mixing is a potential cause of severe data corruption, which can look deceptively correct. By having taken multievent to the extreme, shadow readout is particularly exposed to this. Multicrate systems naturally also have more vulnerable components.

To show that the regular synchronization and event counter checks are effective against unintentional event-mixing due to spurious triggers also in shadow readout multicrate systems, a two-crate system, each with one TDC (CAEN V775), was set up. The TDCs are as usual given the same start signal, and one channel in each TDC is fed the same stop signal, which should give a 1:1 correlation graph for corresponding events [see Fig. 11(a)]. The stop signal is generated from the start with a random delay of 0 to 500 ns.

To inject errors, spurious triggers are injected with about 1 Hz in the slave system and its TDC. This introduces additional events from the slave TDC, which cause following events to be erroneously combined between the two systems, until the next synchronization check.

Potentially bad events are separated from the presumed good ones during analysis by the detection of event counter mismatches between the master and slave systems. This is shown in Fig. 11(a), where no mixed correlations are observed among the presumed good events. The bad events still have a pronounced diagonal, since only events in a readout block after the spurious trigger are affected, while the detection granularity is entire readout blocks.

VIII. Conclusion

Three principle ways to operate VME-based readout systems for multievent capable digital acquisition modules have been described. These schemes are single-event, multievent, and shadow readout. The necessary considerations to describe the performance of these schemes in terms of total system deadtime, based on the timing of individual operations, have been detailed. From these considerations, it is expected that the deadtime ratio for a scheme in which data readout is performed asynchronously to the conversion should converge to the limit of the busy time of the front-end electronics. An implementation of such a scheme was then presented in some detail, and it was shown that the deadtime ratio converges as long as the VME bus has sufficient bandwidth for the total data rate, using either SiCy access or block transfer. Finally, a benchmark of two different systems with shadow readout was shown. In both cases, the shadow readout scheme achieved higher livetime than the alternative readout schemes, even when the latter used faster block transfer modes.

At the time of writing, a shadow readout system has been running for several months at the Aarhus University 5-MV accelerator without problems, and the methods have been successfully used for the IS633 and IS616 experiments at CERN ISOLDE.

Acknowledgment

The authors would like to thank Dr. N. Kurz for the work to provide a stable Linux environment with block transfer modes for the RIO4 SBC. They would also like to thank the Daresbury NPG for providing the MVME Linux environment.

References

[1] *American National Standard for VME64*, ANSI/VITA Standard 1.0-1994 (S2011), 1994.
[2] *CAMAC Model 4300B, 16 Channel, Fast Encoding & Readout ADC (FERA)*, LeCroy, Chestnut Ridge, NY, USA, 1986.
[3] R. S. Larsen, "Introduction to the FASTBUS standard data bus," *Interfaces Comput.*, vol. 1, no. 1, pp. 19–31, 1982.
[4] V. Pucknell and D. Laff. *MIDAS the Multi Instance Data Acquisition System*. Accessed: Oct. 6, 2018. [Online]. Available: http://npg.dl.ac.uk/MIDAS/
[5] A. Kumar, A. Chatterjee, K. Mahata, and K. Ramachandran, "Development of data acquisition software for VME based system," in *Proc. PCaPAC*, Kolkata, India, 2012, pp. 35–36.
[6] *Silena 9418 Acquisition Control module*. Accessed: Apr. 18, 2018. [Online]. Available: http://npg.dl.ac.uk/MIDAS/VME/sac.html
[7] K. Ramachandran, private communication, May 2018.
[8] S. Ritt and P. Amaudruz, "The MIDAS data acquisition system," in *Proc. IEEE 10th Real Time Conf.*, Jul. 1997, pp. 309–312. [Online]. Available: http://daq.triumf.ca/~daqweb/ftp/publications/rt97_paper.ps
[9] H. G. Essel and N. Kurz, "The general purpose data acquisition system MBS," *IEEE Trans. Nucl. Sci.*, vol. 47, no. 2, pp. 337–339, Apr. 2000.
[10] H. Baba *et al.*, "New data acquisition system for the RIKEN radioactive isotope beam factory," *Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc. Equip.*, vol. 616, no. 1, pp. 65–68, 2010.
[11] H. Baba *et al.*, "Parallel readout VME DAQ system," RIKEN, Wako, Japan, RIKEN Accel. Progr. Rep. II-9, 2017, p. 224, vol. 50.
[12] G. F. Knoll, *Radiation Detection and Measurement*, 4th ed. Hoboken, NJ, USA: Wiley, 2010.
[13] mesytec GmbH & Co. KG, Putzbrunn, Germany. *Mesytec TDC Data Sheet V2.6*. Accessed: Aug. 28, 2018. [Online]. Available: https://www.mesytec.com/products/datasheets/MTDC-32.pdf
[14] mesytec GmbH & Co. KG, Putzbrunn, Germany. *Mesytec ADC Data Sheet V2.1*. Accessed: Aug. 28, 2018. [Online]. Available: https://www.mesytec.com/products/datasheets/MADC-32.pdf

[15] *Mod. V785, 16/32 Channel Peak Sensing ADC Technical Information Manual*, CAEN, Viareggio, Italy, Feb. 2012.
[16] *Mod. V1190, Multihit TDC Technical Information Manual*, CAEN, Viareggio, Italy, Nov. 2016.
[17] J. Hoffman. (2013). *VULOM4b Data Sheet*. Accessed: May 10, 2017. [Online]. Available: http://www.gsi.de/fileadmin/EE/Module/VULOM/vulom4B_3.pdf
[18] H. T. Johansson, M. Heil, B. Loher, H. Simon, and H. Törnqvist, "TRLO II—Friendly FPGA trigger control," GSI, Darmstadt, Germany, GSI Sci. Rep. GSI-SR2013-FG-S-FRS-15, 2014, p. 354. [Online]. Available: http://repository.gsi.de/record/68045
[19] *2eSST*, ANSI/VITA Standard 1.5-2003 (S2014), 2003.
[20] *MVME5500 Single-Board Computer Programmer's Reference Guide*, Motorola, Tempe, AZ, USA, Oct. 2003.
[21] *RIO4-8072 User Manual, Version 1.0*, CES, Geneva, Switzerland, Nov. 2009.
[22] (Feb. 2005). *Installation of MIDAS Application Packages for MVME5500 Linux*. Accessed: Sep. 23, 2018. [Online]. Available: http://npg.dl.ac.uk/MIDAS/download/mcglinux.html
[23] N. Kurz and J. Adamczewski-Musch, "GSI data acquisition system MBS release notes V6.2," GSI, Darmstadt, Germany, Apr. 2013.
[24] B. Löher, A. Charpy, H. T. Johansson, H. Simon, and H. Törnqvist, "The DAQ readout library vmelib," GSI, Darmstadt, Germany, GSI Sci. Rep. GSI-SR2014-MU-NUSTAR-NR-08, 2015, p. 192. [Online]. Available: http://repository.gsi.de/record/183940
[25] R. Catherall *et al.*, "The ISOLDE facility," *J. Phys. G, Nucl. Part. Phys.*, vol. 44, no. 9, p. 094002, Sep. 2017.
[26] *User Manual UM3074, Digital Detector Emulator*, CAEN, Viareggio, Italy, Jan. 2016.
[27] CAEN, private communication, Apr. 2018.
[28] J. Hoffmann, N. Kurz, and M. Richter, "TRIVA, VME trigger module, user manual," GSI, Darmstadt, Germany, 2007.